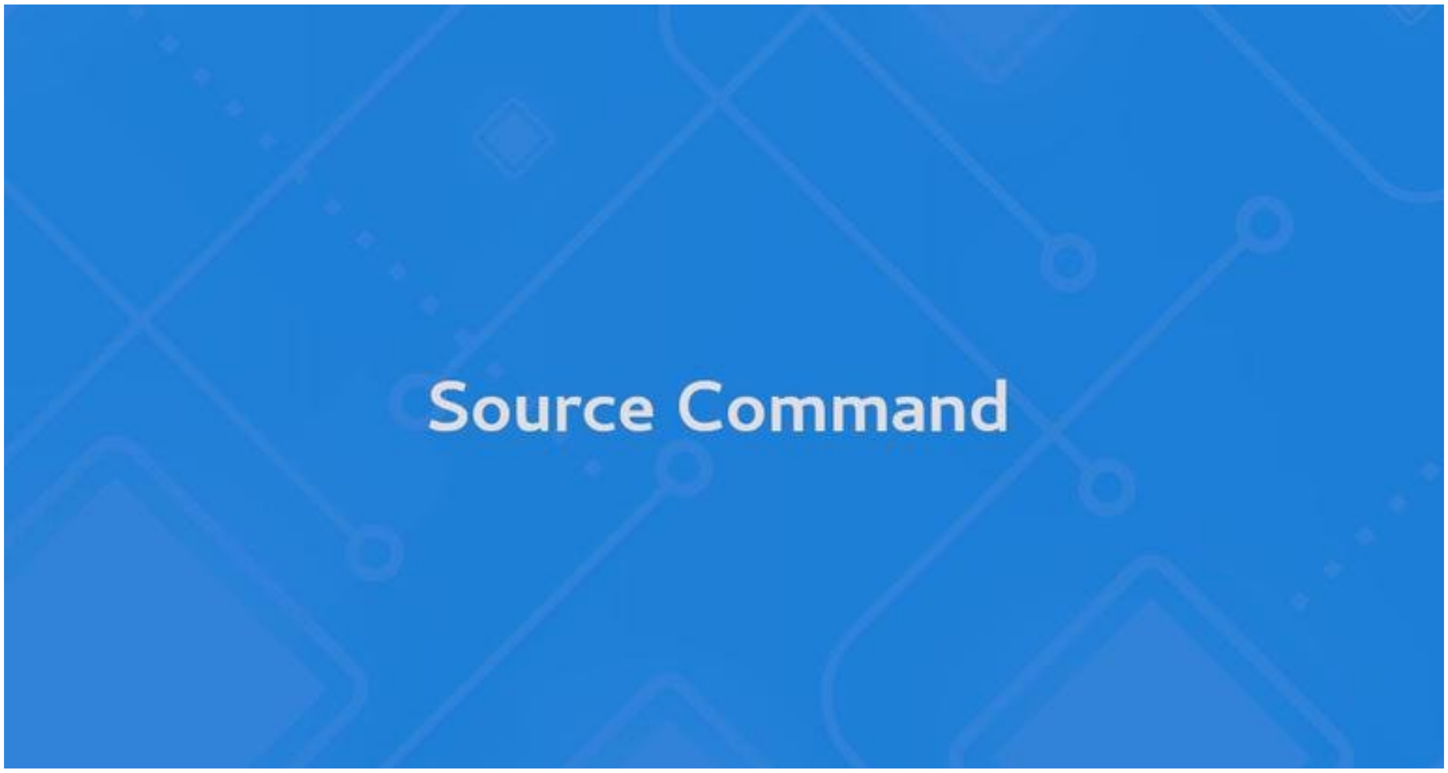
# Bash Source Command

Updated Jun 6, 2020 • 3 min read



The `source` command reads and executes commands from the file specified as its argument in the current shell environment. It is useful to load functions, variables, and configuration files into shell scripts.

`source` is a shell built-in in Bash and other popular shells used in Linux and UNIX operating systems. Its behavior may be slightly different from shell to shell.

## Source Command Syntax

The syntax for the `source` command is as follows:

```
. FILENAME [ARGUMENTS]
```

- `source` and `.` (a period) are the same command.
- If the `FILENAME` is not a full path to a file, the command will search for the file in the directories specified in the `$PATH` environmental variable. If the file is not found in the `$PATH`, the command will look for the file in the current directory.
- If any `ARGUMENTS` are given, they will become positional parameters to the `FILENAME`.
- If the `FILENAME` exists, the `source` command exit code is `0`, otherwise, if the file is not found it will return `1`.

## Source Command Examples

In this section, we will look at some basic examples of how to use the `source` command.

### Sourcing Functions

If you have shell scripts using the same functions, you can extract them in a separate file and then source that file in your scrips.

In this example, we will create a file that includes a bash function that checks whether the user running the script is the root, and if not, it shows a message and exits the script.

functions.sh

```
check_root () {
    exit 1
  fi
}
```

Now in each script that needs to be run only by the root user, simply source the `functions.sh` file and call the function:

```
#!/usr/bin/env bash

source functions.sh
check_root

echo "I am root"
```

If you run the script above as a non-root user, it will print “This script must be run as root” and exit.

The advantage of this approach is that your scripts will be smaller and more readable, you can reuse the same function file whenever needed, and in case you need to modify a function you’ll edit only one file.

## Bash Configuration file

Let’s create a test configuration file:

config.sh

```
VAR1="foo"
VAR2="bar"
```

In your bash script, use the `source` command to read the configuration file:

```
#!/usr/bin/env bash

source config.sh

echo "VAR1 is $VAR1"
echo "VAR2 is $VAR2"
```

If you run the script the output will look like this:

```
Output
VAR1 is foo
VAR2 is bar
```

## Conclusion

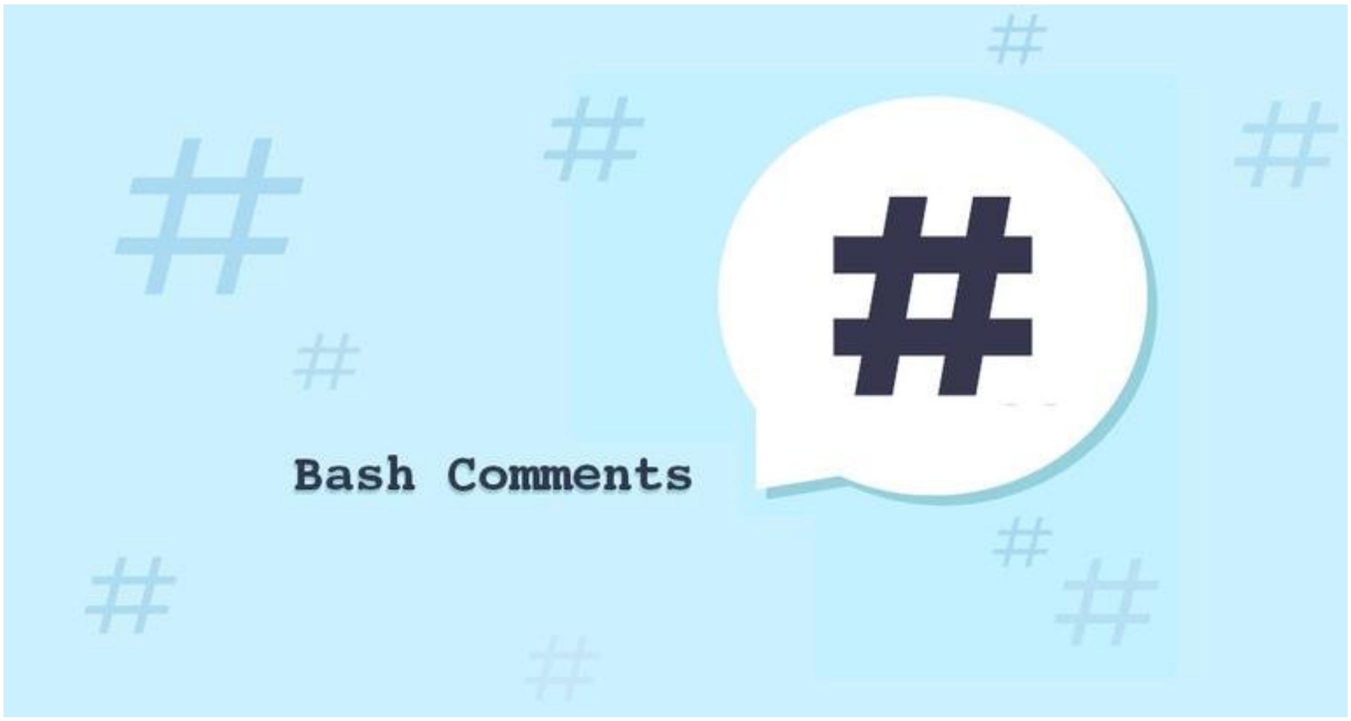
In this guide, you have learned how to use the `source` built-in command in your shell scripts.

If you have any questions or feedback, feel free to leave a comment.

## Related Articles

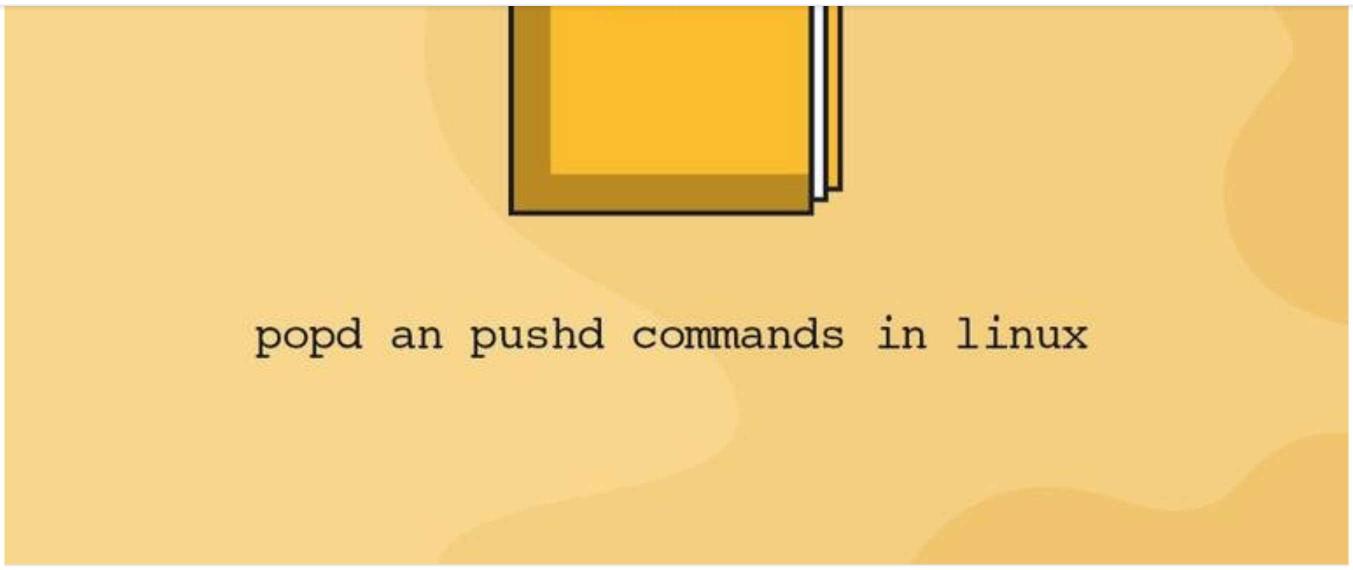
OCT 2, 2019

Writing Comments in Bash Scripts



SEP 21, 2019

Pushd and Popd Commands in Linux



Show comments (3)